
Article:
Almanza Medina, José, Henson, Benjamin and Zakharov, Yury orcid.org/0000-0002-2193-4334 (2021) Deep Learning Architectures for Navigation using Forward Looking Sonar Images. IEEE Access. pp. 33880-33896. ISSN 2169-3536

https://doi.org/10.1109/ACCESS.2021.3061440

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Deep Learning Architectures for Navigation using Forward Looking Sonar Images

JOSÉ E. ALMANZA-MEDINA, (Graduate Student Member, IEEE), BENJAMIN HENSON, (Member, IEEE), AND YURIY V. ZAKHAROV, (Senior Member, IEEE)

Department of Electronic Engineering, University of York, York YO10 5DD, U.K.

Corresponding author: José E. Almanza-Medina (e-mail: jeam502@york.ac.uk)

J. E. Almanza-Medina acknowledges financial support from CONACyT. The work of Y. Zakharov and B. Henson was supported in part by the U.K. EPSRC through Grants EP/P017975/1 and EP/R003297/1.

ABSTRACT This paper investigates the use of supervised Deep Learning (DL) networks to process sonar images for underwater navigation. State-of-the-art DL techniques for micro-navigation using sequences of optical images have been adapted to work with sonar images. Specifically, the DL networks estimate the Forward-Looking Sonar (FLS) motion in three degrees of freedom corresponding to x- and y-translation and rotation around z-axis. The state-of-the-art DL architectures and a proposed new architecture are investigated for motion estimation. They are trained using images generated by a FLS simulator. The data sets are made using pairs of consecutive images associated with labels that represent the motion of the sonar platform between images. The results show the effectiveness of using the DL architectures, which can provide millimeter accuracy for translation motion and below 0.1° for rotation motion between two consecutive sonar images. Examples of trajectory estimation and mosaic building using simulated and real sonar images are also presented.

INDEX TERMS deep learning, trajectory estimation, underwater micronavigation

I. INTRODUCTION

THE use of non-piloted underwater vehicles has become an essential tool in exploration and surveying of underwater environments [1]. Autonomous underwater vehicles (AUVs) and remotely operated underwater vehicles (ROVs) alleviate the dangers that humans are exposed during explorations. Accurate navigation of these vehicles is required to succeed in their tasks [2]. However, navigation underwater is still a challenge and is an active area of research [3]. The navigation problem can be addressed on a large scale (macronavigation) and on a small scale (micronavigation). Technologies for macronavigation such as global positioning system do not work underwater as they do in land-based applications since radiowaves are highly attenuated when passing through water bodies [4]. The use of undersea acoustic beacons can be used for underwater navigation [5], but it also makes the system complexity and cost high [6] and they have a limited operation area [7].

Micronavigation approaches have been implemented for applications underwater. Inertial navigation systems (INS) can be employed underwater but their accuracy is affected by the gyroscope drift and accelerometer bias [8], [9]. Then, INS are used as an aid to other techniques such as Displaced Phase Center Antenna (DPCA) for Synthetic Aperture Sonar (SAS) micronavigation [10], [11]. Alternatives are visual odometry [12], [13] and visual simultaneous localization and mapping (SLAM) [14], [15]. However, visual based navigation methods are unreliable under conditions of poor visibility caused by water turbidity or scarce illumination [16]–[18]. Under these circumstances, acoustic imaging techniques present advantages over optical imaging. Acoustic waves can travel through the water regardless the water transparency and work effectively at larger ranges.

The use of Deep Learning (DL) techniques applied in image, video and audio processing [19], [20] has lead to the solution of complex problems where deterministic and other artificial intelligence techniques have been insufficient. DL techniques for image registration have been developed for such applications as optical flow and ego-motion estimation [21]–[25], displacement in magnetic resonance im-

ages [26], [27] and synthetic aperture radar images [28]–[30].

For underwater scenarios, some effort has been dedicated to apply DL to sonar imaging. However, most of the studies have been focused on object classification [31]–[35]. DL in a non-classification problem applied to sonar images is presented in [36], where an overlap between two sonar images is estimated. However, DL techniques for navigation using acoustic imagery is still an area to be explored.

The work [37] presents a deterministic algorithm for attitude and trajectory estimation of an underwater platform from a data set of sonar images. However, this algorithm is not suitable for a real-time implementation due to the high computation time. Therefore, the aim of this paper is to implement a framework that uses advanced DL architectures for real-time motion estimation from consecutive sonar images. The DL architectures are trained using images generated by a sonar simulator. Then, the estimates are used to reconstruct the navigation trajectory of the platform. Additionally, a mosaic that combines the sonar images is built following the reconstructed trajectory. The DL architectures considered in this paper are based on DL architectures developed for motion estimation using optical images. Analysis of the state-of-the-art techniques from the literature shows that the following architectures are sufficiently advanced to achieve high precision motion estimation: SfMNet [22], PoseNet [23], [24], CNN1b and CNN4b networks [21]. Different from optical images, sonar images are monochrome and have lower resolution [34], making them less informative for the motion estimation. In this paper, we modify these architectures for working with sonar images and investigate the performance of the motion estimation.

From the machine learning point of view, this is a regression problem. When training a neural network for complex regression tasks, a large amount of labeled training data is required [38], e.g., tens of thousands images for optical flow estimation [39], [40]. Manual labeling of training data can be a time consuming task [41]. Furthermore, in some real scenarios, it can be hard to obtain adequate data to train a network [42]. Data augmentation can partly resolve this problem by artificially enlarging the size of the training set whilst keeping the labels [43]. The use of synthetic data generated by computer can be an alternative to alleviate this issue and can be used as a source of large sets of labeled training data [38].

Synthetic data sets made entirely by computers have been used in DL for training and testing purposes, specifically in computer vision applications such as face recognition [44], [45], object detection/classification [41], [42], [46], [47], text detection and recognition [48], [49], captcha recognition [38], etc.

The present work uses a Forward-Looking Sonar (FLS) simulator to generate large volumes of synthetic sonar images. The images are used for training and validation of the DL networks. The process followed in this paper for training the networks is illustrated in Fig. 1 and it consists of the following steps:

1) The sonar simulator generates images while moving in simulated environments and stores the sonar position where each image is generated.
2) Data sets for training and validation are generated using the images and positions. Each training sample is obtained by concatenating a pair of consecutive images into one single image. The label corresponds to three motion parameters required to move from the position where the first image is acquired to the position where the second image is acquired as detailed in Section III. The labels are quantized and normalized as described in subsections IV-A and IV-B, respectively. A data sample consists of a concatenated image and a label.
3) The DL network architecture is defined. Five different architectures are considered as described in Section III.
4) The data set is split into a training set and a validation set (95% and 5% of the whole data set, respectively). For a fair comparison, each network is trained using the same data.
5) The network is trained by using as input the training samples.
6) Once the network is trained, the root mean squared error (RMSE) for each of the three parameters is calculated.

The trained networks are used to estimate the motion of the sonar platform from a simulated or real data set obtained along a trajectory. This process is illustrated in Fig. 2 and described as follows:

1) An already trained network is chosen to estimate the motion parameters from sonar images in the data set.
2) Pairs of consecutive images in the data set are concatenated into a single image. There are no training labels since the network is used for estimation.
3) The concatenated images are applied to the network, which produces estimates of three motion parameters for every concatenated image.
4) The trajectory of the sonar is computed as described in subsection V-A.
5) Optionally, a mosaic can be built by merging the sonar images according to the estimated trajectory.

The remainder of this paper is organized as follows. Section II describes the data sets and sonar parameters employed for training and validating the networks. Section III describes the DL architectures used for ego-motion and trajectory estimation and modifications made to work with the sonar images. Section IV presents a performance comparison of the DL architectures. Section V describes the procedures and presents results of trajectory estimation and mosaic building using synthetic and real sonar images. Conclusions are given in Section VI.

II. SONAR APPLICATION SCENARIOS AND DATA SETS

In this section, four sonar image data sets are described. Two of these data sets are used for training and validating the network. They are described in detail in subsections II-A

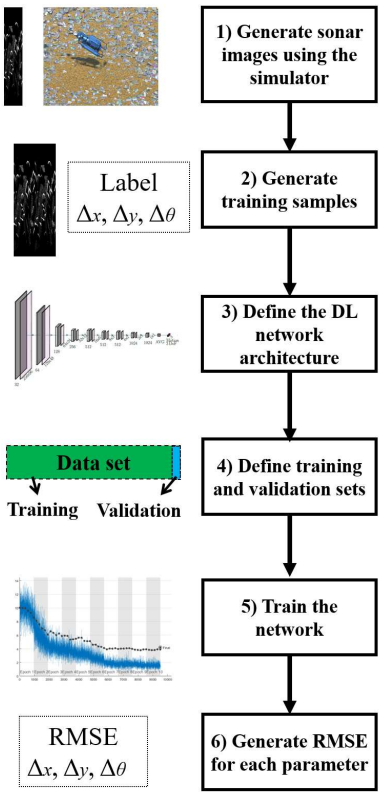


FIGURE 1: Data flow diagram of the process for training the DL networks with data generated by the simulator.

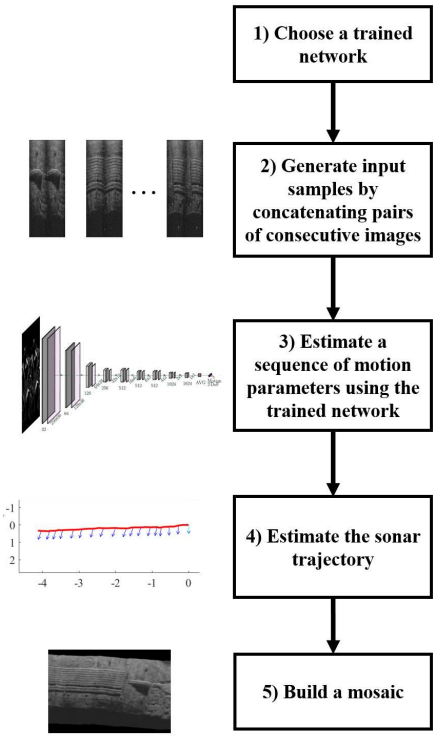


FIGURE 2: Data flow diagram of the process for motion and trajectory estimation using a trained network.

and II-B. Both data sets are made of synthetic images generated using the Unity engine sonar simulator [50]. The sonar simulator is based on a ray-tracing technique. Examples of simulation scenarios used for each data set are shown in Fig. 3. The first one represents an underwater bottom field covered with rocks, whilst the second one represents the surface of a ship's hull (Fig. 3a and 3b, respectively).

When generating data from a simulator, there always is a modeling error, resulting in a reduced accuracy of the estimator. The simulator parameters should be chosen to guarantee a minimum of this error. With the sonar simulator, such parameters as the level of noise, size and reflectivity of objects in the simulated environments should be adjusted to match the real scenarios where the estimator is going to be used. The accuracy of the estimator can be improved using training and validation with a combination of simulated and real data. However, labeling the real data is a complicated problem. The accuracy in real scenarios can also be improved with sufficiently high variability of the simulated data. The ultimate test of an estimator should be preferably done using real data. In this paper, we use the sonar simulator to generate a highly variable data set for training the networks and further test the motion estimator using real data.

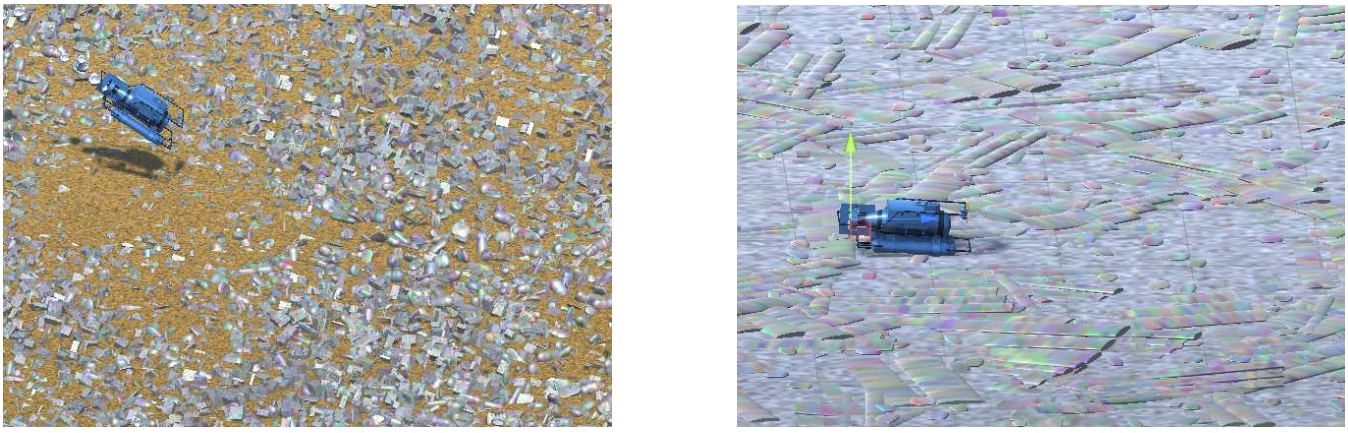
The simulated sensor is the DIDSON 300 sonar [51]. It has a Field of View (FoV) of $29^\circ \times 14^\circ$ (azimuth and elevation angles, respectively), 96 beams in the azimuth dimension, image size of 512×96 pixels and intensity range between 0 and 255. The frame rate is 21 frames (images) per second (fps). The images are generated without image noise, which is added later for validation and network performance enhancement purposes.

Two more data sets are described in subsections II-C and II-D. They consist of images acquired from real sonar sensors in the inspection of a ship's hull and a dam wall, respectively.

A. SIMULATED ROCKY FIELD DATA SET

For the rocky field data set, thirty different scenarios were created and 200000 images generated. Each scenario has a flat seabed, where three different types of geometrical objects are placed to simulate rocks: cubes, capsules and cylinders. Ten scenarios have an area of 30×30m and twenty scenarios have an area of 50×50m. From each of the first ten scenarios, 5000 images were generated and from each of the other twenty scenarios 7500 images were generated.

The difference between scenarios is in the position and the number of elements of each type of rock. The number of elements is defined by a square grid of size p by side, where p is an integer number in the range from 30 to 130. A rock is placed on each vertex of the grid. The value of p is different for the cubes, cylinders and capsules. Therefore, there are 3 different values of p for each scenario. Also, p values change from scenario to scenario. Each rock is slightly shifted from its corresponding vertex by values ψ and ξ on the



(a) (b)

FIGURE 3: (a) Simulated rocky bottom field underwater scenario. (b) Simulated ship's hull surface scenario.

xy-plane of the grid, respectively. Both values are randomly generated for each single rock using a uniform distribution within the grid step. The height from the seafloor is a random value between 0 and 0.45m. The size of a rock in each of the three dimensions is a random value uniformly distributed in the range from 0 to 0.45m. The rocks are rotated around each axis by a random number in the range of -165° and 100°. The rotation value is independent for each axis.

In this sonar simulator, the acoustic reflectivity of the objects is coded by a number between 0 and 1, as described in [50]. The reflectivity is generated randomly by dividing the whole seabed into a grid of 2048 by 2048 cells, where each of the grid cells has a random value chosen from a uniform distribution. For the rocks, the reflectivity on their surface is assigned according to an image generated using the Perlin algorithm for creating realistic random images [52].

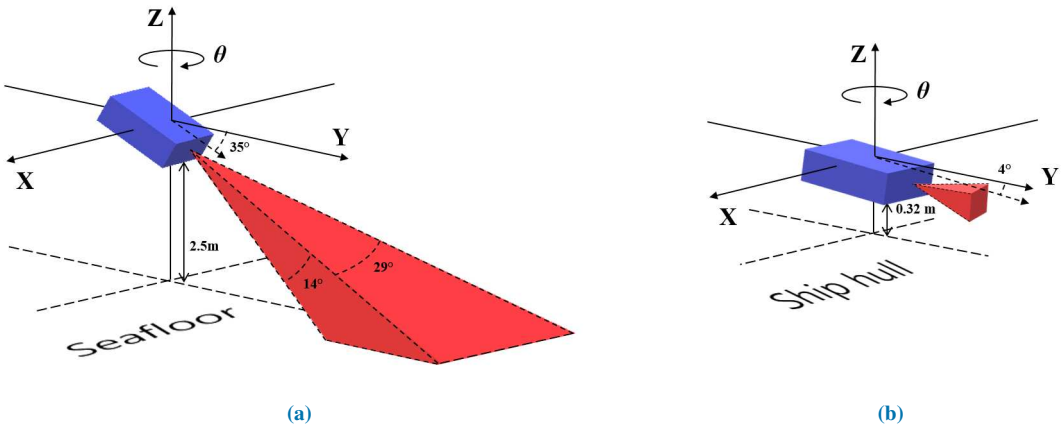
The simulated sonar is looking down with a rotation angle around x-axis of 35° (see Fig. 4a). The height from the seafloor is fixed at 2.5 m. The simulator generates a large number of sequences of 5 images. For each sequence the sonar moves at a constant rate in each DoF; the rate is defined randomly with a uniform distribution within limits of the maximum motion and rotation speed of the sonar platform. The limits for x and y-translations are $\pm$ 420mm/s and the limit for rotation around the z-axis is $\pm$ 9.45°/s (the maximal displacement of $\pm$ 20mm and rotation of $\pm$ 0.45° between consecutive images, respectively). For deciding on the translation limits, the specification of the Bluefin Robotics Hovering Autonomous Underwater Vehicle [53] is referenced along with the real data set described in subsection II-C. For the rotation, the maximum speed was chosen focusing on low rotation navigation, such as ship's hull inspection.

After generating a sequence of 5 images, the sonar is randomly moved to a different place into the simulated scene. It is displaced within 3 to 4m in each of the x and y-axis, while the rotation around the z-axis is a uniform random value in the range between $-$ 180 and 180°. This large displacement is to avoid over-generating images of the same view and area. After moving to a new place of the scenario, the next sequence of 5 images is generated using new motion rates. In total 40000 sequences of five images were created. The data set is made by concatenating pairs of consecutive images from each sequence. Therefore 4 pairs are made from each sequence. The total number of pairs of images in the data set is 160000.

The data set labels correspond to the displacement of the sonar platform from the position where an image is obtained to the position of the next image. Each label has three parameters that correspond to x- and y-translation and rotation around z-axis. The data set was shuffled and divided into training and validation sets with 95% and 5% of the data, respectively.

B. SIMULATED SHIP'S HULL DATA SET

Fifteen scenarios were created to simulate the bottom of a ship's hull. In each scenario, a flat surface with two different types of objects is simulated: (i) groups of flat tubes aligned one beside the other and (ii) small flat cylinders that represent sacrificial anodes. The objects are attached to the ship's hull surface. They can be seen in Fig. 3b. The tubes are generated as elliptical cylinders. All the sizes are independent uniformly distributed random variables in the ranges as follows. The cross-section of a tube is an ellipse with the major axis in the range from 0 to 0.4m and the minor axis in the range from 0 to 0.05m. The length of a tube is between 1 and 3m. The anodes are also elliptical cylinders. Their cross-section is facing up. The height is in the range from 0 to 0.03m. The major and minor axis are in the range from 0.15m to 0.3m and from 0.05m to 0.1m, respectively. The position of an object is generated as in the rocky field scenarios. The hull surface reflectivity is generated by dividing the whole surface into a grid of 1024 $\times$ 1024 cells. A random value is assigned to each cell using a uniform distribution between 0.25 and 1. The reflectivity on the surface of the tubes and



(a) (b)

FIGURE 4: Coordinate system of the sonar platform used for generating the data sets. Forward/backward motion of the platform corresponds to the y-axis, whilst sideway motion corresponds to x-axis. The height from the seafloor is constant. Rotation around the z-axis corresponds to the parameter θ. (a) Sonar height and pitch of 2.5 m and 35°, respectively, used in the rocky field scenarios. (b) Sonar height and pitch of 0.32 m and 0.4°, respectively, used in ship's hull scenarios.

anodes is defined with the Perlin algorithm. The simulated sonar height from the hull and pitch angle are fixed at 0.32m and 4°, respectively (see Fig. 4b). The height and pitch were chosen based on estimates from [37] for a real ship's hull data set. The sonar translation and rotation speed limits are the same as that in the rocky field scenarios. From the fifteen scenarios, a total of 100000 images were generated. Then from each sequence of 5 images, 4 pairs of images are made to make a total of 80000 pairs of images in this data set.

C. REAL SHIP'S HULL DATA SET

This data set was obtained using a Bluefin Robotics Hovering Autonomous Underwater Vehicle [53] equipped with a DIDSON 300 sonar as described in [54]. The data set consists of a sequence of 4464 images that show the inspection of a ship's hull. A subsequence of 520 images is extracted. The subsequence represents one single pass from end to end of the ship, for a total length of around 10 meters.

D. REAL DAM INSPECTION DATA SET

This data set is from the inspection of a dam wall [55], [56]. It consists of 1596 images obtained using the ARIS 3000 sonar [57] while doing a single pass along the dam wall, moving principally in the sideways direction.

III. DL NETWORKS FOR ATTITUDE-TRAJECTORY ESTIMATION

In this section, five DL networks for trajectory estimation are described. The first four networks are state-of-the-art networks for trajectory estimation using optical images and the fifth one is a new architecture.

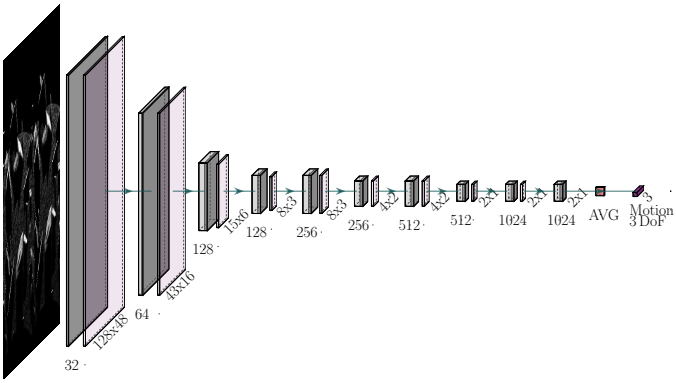
Six degrees of freedom (DoF) are needed to represent the motion of a sonar sensor. They correspond to the translations in $x-$, $y-$ and $z-$ axes and the rotation around each axis. We assume that the height from the seafloor is constant for the duration of the sonar exploration and that rotations around $x-$ and $y-$ axes are negligible. Therefore, for this work the DL networks were adapted to work with sonar images for motion estimation between a pair of images in 3 DoF: $\Delta = [\Delta_x, \Delta_y, \Delta_\theta]$, translation on the xy-plane parallel to the seabed and rotation around the z-axis (denoted by θ), respectively (see Fig. 4).

The network architectures are shown in Fig. 5. The input of all the networks is a data set of images obtained by concatenation of a pair of images. Since each image size is 512×96 pixels, the size of the concatenated input image is 512×192 pixels.

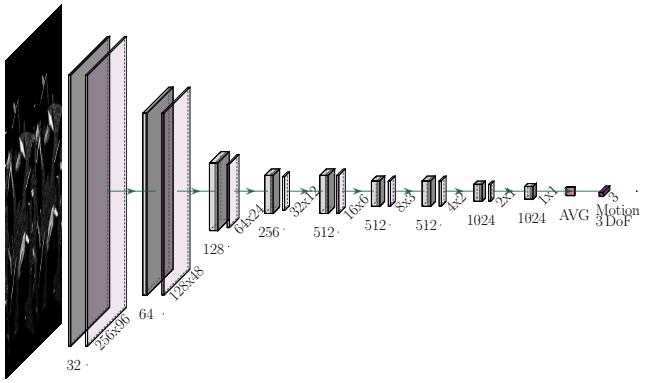
A. SFMNET

SfMNet [22] is a self-supervised DL network designed to estimate the camera motion from a sequence of images (rotation and translations). The architecture of the SfMNet is divided into two sections: the first section uses convolutional layers to estimate the camera motion and the second section uses deconvolutional layers to obtain a motion mask that is used to generate a pixel motion estimate. For this paper, we are only interested in the first section. The network architecture is shown in Fig. 5a. The architecture from [22] is repeated as much as possible but some modifications are required to deal with the sonar images. In this work, ten convolutional layers are used to build the network; only one of the 64 channels layer from the network in [22] is used and the other is removed. The network has shown better results and lower training time without this layer.

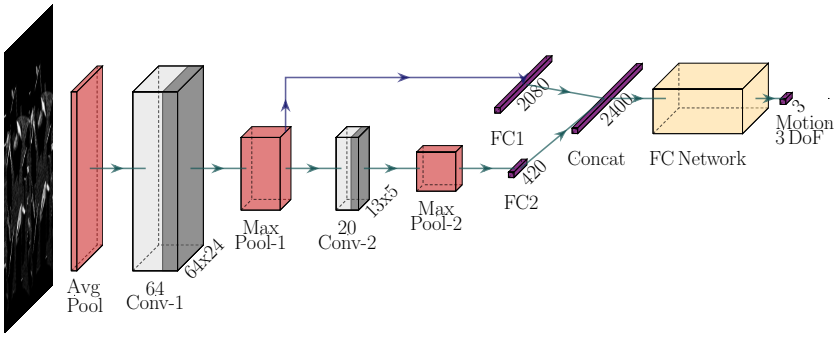
The kernel size of the convolutional layers is 3×3, except for the first two layers, where the kernel size is 7×7 and 5×5, respectively. The stride of the first three layers is 4, 3 and 3, respectively. The rest of convolutional layers have a stride that alternates between 2 and 1 in each layer and the number of channels increases by two every two layers. A batch normalization layer and a ReLU activation function are placed after each convolutional layer, except for the last one



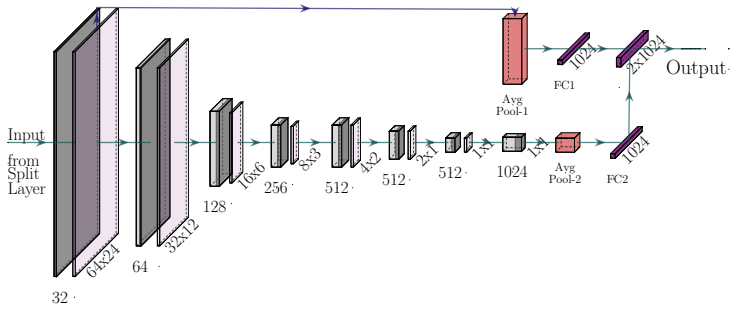
(a) SfMNet - Motion estimation section



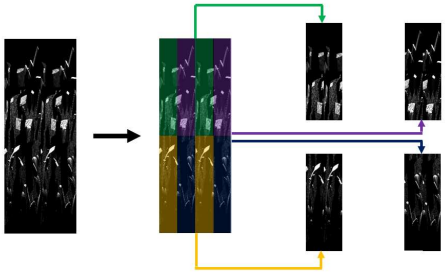
(b) GeoNet - PoseNet



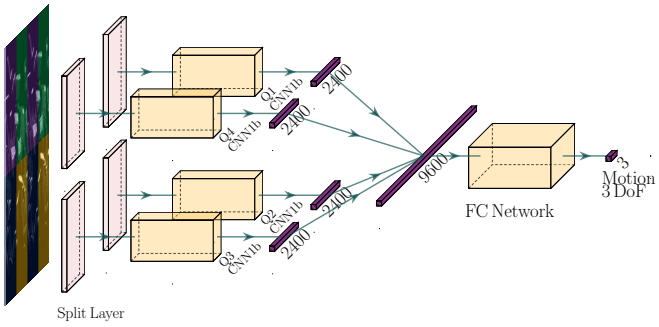
(c) CNN-1b



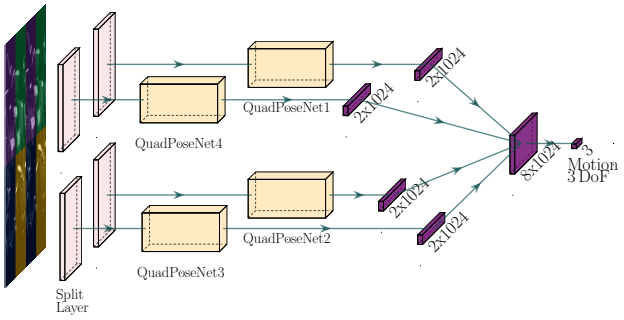
(d) QuadPoseNet



(e) Split layer



(f) CNN-4b



(g) CNN-4QuadPoseNet

FIGURE 5: DL architectures for estimation of 3 DoF of position and orientation of an underwater platform using acoustic images.

that is connected to an average pooling layer. The average pooling layer improves the performance when images are noisy. The final layer is a regression layer that outputs the 3 estimated DoF. Dropout regularization with a rate of 50% is applied when training the network to prevent overfitting.

B. GEONET: POSENET

GeoNet [24] is also a self-supervised DL architecture that can perform ego-motion estimation from videos. The full network architecture consists of three subnetworks called DepthNet, PoseNet and ResFlowNet. For our work, we are only interested in the PoseNet architecture (position and orientation estimation). The PoseNet is based on the architecture from [23]. This network receives a sequence of N images as input, then 7 convolution layers with stride 2 are applied, followed by a 1×1 convolution layer with $6 \times (N-1)$ channels, six DoF are estimated by comparing one of the N images with the others. All the convolutional layers, except the last one, are followed by an ReLU activation function and batch normalization. An average pooling layer is applied before the output regression layer to generate the motion estimates. This architecture was implemented and trained. However it did not show any learning progress when applied to the sonar images. Therefore it was modified by adding extra convolutional layers to make a total of 9 such layers as shown in Fig. 5b, where the number of channels in each convolutional layer can be seen below each block. Dropout regularization with a rate of 50% is applied during the training. In this work, the output regression layer size is $3 \times (N-1)$, since it estimates three DoF. The number of input images is $N = 2$.

C. CNN-1B AND CNN-4B

In [21], a DL framework with three types of CNNs to determine the trajectory of a camera from multiple images is presented. For our work, two of the three CNNs are implemented and compared: CNN-1b and CNN-4b. The network parameters of the CNN-1b are adjusted to work with sonar images. The images are smaller than the optical images used in this reference. Therefore, the internal layers of the network are adjusted accordingly. The adjusted CNN1-b can be seen in Fig. 5c. It takes an image and downsamples it with an average pooling layer with kernel of 8×8. It reduces the number of parameters to train, and, as a result, the computational cost. After downsampling, the layers are as follows: a convolutional kernel of size 9×9, a 4×4 max pooling layer, another convolutional kernel of size 2×2 and a 2×2 max pooling layer. The outputs of each max pooling layer are concatenated into one single fully connected layer that is then connected to a fully connected network, made of 4 fully connected layers to obtain the motion estimate.

The parameters of the network CNN-4b are adjusted to to match the size of the sonar images. The adjusted network is shown in Fig. 5f. It uses a Split layer (Fig. 5e) to segment each of the concatenated images into 4 quadrants, then the corresponding quadrants are concatenated to make a sub-image. Each sub-image is downsampled 4 times by an average pooling layer. CNN-1b is then applied on each sub-image. The final layer combines the outputs of the four CNN-1b networks. The fully connected network takes the outputs of the concatenated layers as the input. The sizes of the fully connected layers are 2400, 1200, 600 and 300, respectively. The final layer of the network is the regression layer.

D. CNN-4POSENET

A new architecture is proposed to exploit the segmentation into quadrants implemented in the CNN-4b network and the high number of layers of GeoNet-PoseNet. The complete network is displayed in Fig. 5g. The input of the network is the pair of concatenated images. Similarly to the CNN-4b network, the images are split into 4 sub-images and the quadrant of one image is concatenated with the corresponding quadrant of the other image. Then each of the 4 concatenations is passed to a subnetwork called QuadPoseNet. This subnetwork consists of 8 convolutional layers with a ReLU activation function and a batch normalization layer. The convolutional layers use the same kernel size of the convolutional layers as the PoseNet described above. After the last convolutional layer, an 4×4 average pooling layer is used, whose output applies to a fully connected layer of size 1024. Additionally, the output of the first ReLU function after the first convolutional layer is connected to a 5×5 average pooling layer. Then this layer is connected to another fully connected layer of size 1024. The two fully connected layers of size 1024 are combined into one single fully connected layer of size 2×1024 to make the output of each QuadPoseNet. Combining layers from the beginning and end of the series of convolutional layers follow the approach of the original CNN-4b to combine coarse and fine features from the images to perform the estimates. The QuadPoseNet layers can be seen in Fig. 5d. The outputs of the four QuadPoseNets are concatenated into another fully connected layer of size 8×1024 and then the final regression layer to compute the motion estimates.

IV. PERFORMANCE ANALYSIS OF THE NETWORKS

A. MOTION ESTIMATION USING SIMULATED DATA

The pairs of images in polar coordinates are the input to the networks. The labels are given by the simulator in meters for the displacements and degrees for the rotation. The three parameter labels are transformed into parameters measured in the same units by a quantization process using the minimum and maximum values of each type of label. Five bits of resolution are used to define the quantization levels, i.e., the labels are assigned to one of 32 values (0 to 31).

A regression output layer with the Mean Squared Error (MSE) loss function is used to measure the estimation error during the training. The loss function $\mathcal{L}$ is given by

$$\mathcal{L} = \frac{1}{2SR} \sum_{k=1}^{S} \|\Delta_k - \Gamma_k\|^2,$$

where $\Delta_k = [\Delta_{x_k}, \Delta_{y_k}, \Delta_{\theta_k}]$ are the estimates for each DoF, $\Gamma_k = [\Gamma_{x_k}, \Gamma_{y_k}, \Gamma_{\theta_k}]$ are the quantized labels for each DoF, k is the index that refers to the training examples in the mini-batch, S is the mini-batch size and R is the number of parameters to estimate, which is $R = 3$.

To speed up the training, the data set is split into mini-batches ($S = 4$). The learning rate at the start of the training is set to 0.0001 and it is reduced after 12 epochs. The networks are trained until the validation loss converges. This takes less than 32 epochs depending on the network. The Adam optimization algorithm is used for training.

The five networks are trained using noiseless images. The validation is performed using the validation set with noise and without noise. Noise with parameters measured from real images in [58] are applied to the pixel intensity levels. The pixels that correspond to the acoustic shadows in the images are distorted by additive Gaussian noise, with the mean 35 and a standard deviation of 8. Pixels of objects are modified using an additive noise with the Rayleigh distribution that represents the scattering noise from the surface of the objects. The Rayleigh distribution has a scale parameter of 35.

The validation results for all the architectures are shown in Table 1. It can be seen that the architectures with the best performance, when applied to noiseless images, are the PoseNet and the proposed CNN-4PoseNet, achieving a similar navigation RMSE in translation in x-direction and rotation around z-axis. The RMSE in translation in x-direction is slightly smaller for PoseNet whilst in translation in y-direction, it is smaller for CNN-4PoseNet. Validation using noisy images shows that the proposed CNN-4PoseNet is less sensitive to the noise than the other networks. CNN-1b and CNN-4b architectures are the most sensitive to the noise.

Also, it can be seen that all the architectures achieve better estimates in the y-axis, corresponding to the forward/backward motion of the platform. When working with images produced by sonars with a small azimuth FoV, it is more difficult to estimate the sideway motion of the platform since rotation around z-axis and x-translation result in very similar distortions in images. The forward/backward movement of the sonar means that the pixel motion is mostly in the range axis, thus making it largely independent of the pixel motion caused by sideway movement and rotation around the z-axis.

To measure the similarity of the estimated parameters, the cross-correlation of the estimation errors is calculated. The PoseNet estimates were used to compute the cross-correlation. The cross-correlation values obtained are ρ_{yx}=0.025, $\rho_{y\theta}$=0.220 and $\rho_{x\theta}$=0.822, for the cross-correlation between translation motions, the motion in y-direction and the rotation and the motion in x-direction and the rotation, respectively. From these values, it can be seen a high correlation between errors obtained for estimates of the sideway movement of the sonar and its z-rotation.

The SfMNet, PoseNet and CNN-4PoseNet architectures were selected for further modifications to try to further reduce the RMSE of motion estimation. They were chosen since they achieved the lowest RMSE among the architectures. The following subsections present results of these attempts.

TABLE 1: Validation errors of the networks.

Approach	RMSE of motion estimation		
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(deg)*
	Images without noise		
SfMNet	4.86	2.36	0.084
PoseNet	2.81	1.44	0.049
CNN-1b	7.63	4.63	0.172
CNN-4b	4.22	2.27	0.089
CNN-4PoseNet	3.18	1.15	0.050
	Images with noise		
SfMNet	5.43	3.05	0.098
PoseNet	11.32	6.42	0.199
CNN-1b	13.67	10.60	0.266
CNN-4b	12.41	13.26	0.291
CNN-4PoseNet	4.87	2.63	0.083

B. NORMALIZATION OF THE LABELS

Normalization is applied to the labels of the training and validation data, rather than using the quantization. The labels are normalized to their maximum values, defined by the limits to the displacement and rotation. It was observed that when normalizing the motion parameters to the range from -1 to 1, the networks are not capable of learning during the training. However, when normalizing the motion parameters to lie in the range from -10 to 10, the networks can learn and obtain better estimates than the quantized-labels approach for some parameters. The benefit of using the range from -10 to 10 can be related to the magnitude of the weight values when the internal layers of the networks are randomly initialized.

A comparison between the quantization, and normalization $\times 10$ is presented in Table 2. For PoseNet, the validation using noiseless images shows better estimates for y-translation when using the normalization and almost the same estimation errors for x-translation and z-rotation. For noisy images, a large improvement can be seen in the estimates when using the normalization approach. This makes the normalization of labels more preferable than the quantization.

The SfMNet with the normalization achieves better estimates than with the quantization for all the parameters when images are without noise. For noisy images, the normalization results in slightly higher estimation errors when compared with the quantization.

For CNN-4PoseNet, the normalization results in increasing all the estimation errors.

From the results obtained, the PoseNet with normalization $\times 10$ (PoseNet-Normx10) and CNN-4PoseNet with quantization (CNN-4PoseNet-Qua) are selected to apply further modifications in their training strategy to reduce the estimation error.

TABLE 2: Validation errors for SfMNet, PoseNet and CNN-4PoseNet with quantization and normalization.

Approach	RMSE of motion estimation		
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(deg)*
	Images without noise		
SfMNet-Qua	4.86	2.36	0.084
SfMNet-Normx10	4.16	2.14	0.073
PoseNet-Qua	2.81	1.44	0.049
PoseNet-Normx10	2.84	1.00	0.052
CNN-4PoseNet-Qua	3.18	1.15	0.050
CNN-4PoseNet-Normx10	5.35	1.72	0.079
	Images with noise		
SfMNet-Qua	5.43	3.05	0.098
SfMNet-Normx10	5.55	3.47	0.106
PoseNet-Qua	11.32	6.42	0.199
PoseNet-Normx10	6.54	3.39	0.170
CNN-4PoseNet-Qua	4.87	2.63	0.083
CNN-4PoseNet-Normx10	10.21	4.23	0.239

C. TRAINING WITH NOISY IMAGES

The rocky data set images were modified by adding a low level of noise and they were then used for training the PoseNet-Normx10. The noise added to the images is Gaussian with a mean of 10.2 and standard deviation of 5.1, which correspond to 4% and 2% of the maximum value of intensity of a pixel (255), respectively. The percentages are chosen to alter the images with a low level of noise only. This Gaussian noise is applied to pixels in acoustic shadows. As described in [50], the noise applied to pixels of the objects in the images has a Rayleigh distribution. The scale parameter of the distribution is 10.2 (4% of the maximum value of a pixel intensity). This approach is validated using noiseless images and images with the higher level of noise as described in subsection IV-A.

The results presented in Table 3 show that for both, CNN-4PoseNet-Qua and PoseNet-Normx10, training without noise is slightly better for noiseless images. When validating with noisy images, the PoseNet-Normx10 shows better estimates by the network trained with noisy images. The CNN-4PoseNet-Qua trained with noisy images obtains similar results to training with noiseless images. The results from PoseNet-Normx10 suggest that the level of noise in training images should be considered when the networks are applied to real data.

Based on the results obtained when training with quantization, normalization and using noisy images, the PoseNet-Normx10 is selected to continue with further modifications to reduce the estimation error. The CNN-4PoseNet is discarded given that it shows smaller improvement. Also, the CNN-4PoseNet is less suitable for a real-time implementation since its training and testing times are between 2 and 3 times higher compared to the PoseNet.

TABLE 3: Validation errors when training the PoseNet-Normx10 using images with noise and without noise.

Approach	RMSE of motion estimation		
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(deg)*
	Images without noise		
PoseNet-Normx10	2.84	1.00	0.052
PoseNet-Normx10wNoise	2.94	1.26	0.056
CNN-4PoseNet-Qua	3.18	1.15	0.050
CNN-4PoseNet-QuaNoise	5.00	2.66	0.081
	Images with noise		
PoseNet-Normx10	6.54	3.39	0.170
PoseNet-Normx10wNoise	3.63	1.50	0.070
CNN-4PoseNet-Qua	4.87	2.63	0.083
CNN-4PoseNet-QuaNoise	5.71	1.61	0.065

D. CONCATENATION OF 3 IMAGES

Rather than using two concatenated images as input, 3 images are concatenated and used for training the PoseNet-Normx10. The basis of this is to give more information to the network and thus reduce the estimation error. From each sequence of 5 images from the rocky data set, 3 training samples are generated by concatenating 3 consecutive images, resulting in a total of 90000 samples. The label is the 3 DoF of motion of the platform that corresponds to the constant motion rate for each sequence. The results are shown in Table 4. The 3-image approach has an improvement in the estimation accuracy and higher robustness to noise in images in both axes for translation motion compared to the 2-image approach. However the error in rotation estimation when using noisy images is increased.

TABLE 4: Validation errors for concatenation of 2 and 3 images in PoseNet-Normx10.

Approach	RMSE of motion estimation		
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(deg)*
	Images without noise		
Using 2 images	2.84	1.00	0.052
Using 3 images	2.70	0.87	0.046
	Images with noise		
Using 2 images	6.54	3.39	0.170
Using 3 images	4.96	1.38	0.227

E. USING WIDER IMAGES

The sonar parameters were modified to have 512 beams and a FoV of $60° \times 12°$ (azimuth and elevation angles, respectively). Having more beams and therefore more information in the sideway motion is expected to reduce the estimation error for motion along x-axis and z-rotation. A small data set of 40000 pairs of images was generated. The images were

(a) (b) (c)

FIGURE 6: Concatenated pairs of sonar images with different azimuth FoV, number of beams and type of concatenation. (a) 29°/96 beams - horizontal concatenation. (b) 60°/512 beams - horizontal concatenation. (c) 60°/512 beams - vertical concatenation.

TABLE 5: Validation errors for 29° and 60° azimuth FoV images with different ways of concatenation for PoseNet-Normx10.

Input to the network	**RMSE of motion estimation**					
	Images without noise			*Images with noise*		
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(deg)*	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(deg)*
Original Images - 29°/96 beams	2.75	1.71	0.053	5.92	4.16	0.120
Wide Images - 60°/512 beams - Horizontal concat	2.38	2.55	0.054	9.17	9.21	0.210
Wide Images - 60°/512 beams - Vertical concat	2.39	2.53	0.057	8.93	8.92	0.206

obtained using the first 10 rocky field scenarios of 5000 images each. To generate the images, the sonar follows the same positions as for generation of the original rocky field data set. We have considered two types of concatenation of images, vertical and horizontal, i.e., one image beside the other and one image above the other, respectively. The purpose of this is to investigate if the network can achieve a better result depending on the way that two images are concatenated. The labels are set according to the movement of the sonar between images as before. For visualization, examples of pairs of concatenated images for the original data set (29°/96 beams) and the horizontal/vertical concatenated wider images data set (60°/512 beams) are presented in Fig. 6.

The network used for training is the PoseNet-Normx10. The results obtained can be seen in Table 5. For a fair comparison when using the original images, the network was re-trained using only the first 40000 pairs of the data set. The results show that wider images provide slightly better sideway estimation with noiseless images than the original ones but there is an increase in the error for the forward/backward motion estimation. The error is higher compared to the original network when validating with noisy images. The higher errors for the wider images with noise can be due to the fact that wider images have a larger area with black pixels (with no information). Since the network is trained using noiseless images, when noise is added to images for validation, it produces a larger area with random pixel values in the images with FoV of 60°, compared to the images with FoV of 29°, which may lead to poorer estimates. Also, it is observed that the type of concatenation does not influence the estimation result.

F. REGRESSION LAYER WITH PENALIZATION

The regression layer was modified to penalize estimates in the cases where they fall outside the predefined range of the motion parameters. Each DoF can be penalized independently. The equation that describes the penalty for each DoF is

$$P_{j_k} = \begin{cases} \gamma(\Delta_{j_k} - B_1)^2 & \Delta_{j_k} < B_1 \\ 0 & B_1 \leq \Delta_{j_k} \leq B_2 \\ \gamma(\Delta_{j_k} - B_2)^2 & \Delta_{j_k} > B_2 \end{cases},$$

where $j = x, y$ or θ. γ is a penalization parameter set to 100. B_1 and B_2 are the boundaries defining the non-penalization interval and they are set to -10 and 10, respectively, since the PoseNet-Normx10 architecture is used. Then, the equation for the loss with penalization $\mathcal{L}_\mathcal{P}$ is

$$\mathcal{L}_\mathcal{P} = \frac{1}{2SR} \sum_{k=1}^{S} \left(\|\Delta_k - \Gamma_k\|^2 + P_{x_k} + P_{y_k} + P_{\theta_k} \right).$$

The network is trained using the rocky fields data set and the resulting RMSE for each DoF is shown in Table 6. There is a small difference between using and not using penalization for noiseless images but it is still better not to use the penalization. For noisy images, there is an improvement in translation estimation accuracy but the rotation estimation accuracy is worse. In general, introducing such a penalty does not provide significant improvement.

TABLE 6: Validation errors for the PoseNet-Normx10 when trained with and without penalization.

Approach	RMSE of motion estimation		
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(deg)*
	Images without noise		
Without penalization	2.84	1.00	0.052
With penalization	3.02	1.09	0.051
	Images with noise		
Without penalization	6.54	3.39	0.170
With penalization	5.80	2.67	0.266

G. TRANSFORMATION OF POLAR TO CARTESIAN COORDINATES

The images from the rocky field data set are transformed from polar to cartesian coordinates. The purpose of this transformation is to reduce the sideway RMSE assuming that the higher error is due to the correlation with the rotation. The images in cartesian coordinates are used to train the PoseNet-Normx10.

Additionally, a network that combines images in cartesian and polar coordinates is designed. The combined network takes pairs of images in each coordinate system at the same time and trains a PoseNet-Normx10 network for each type of coordinates. The outputs of both subnetworks are concatenated to a fully connected layer before obtaining the final estimate. The validation results can be seen in Table 7. The RMSEs obtained using cartesian coordinates are slightly higher than the ones obtained with polar coordinates. This could be due to the interpolation error when transforming from polar to cartesian coordinates. However, the results for the combined network show some improvement in estimation of each parameter, especially for noisy images.

H. COMPARISON WITH A DETERMINISTIC METHOD FOR ATTITUDE-TRAJECTORY ESTIMATION

The DL approach used in this paper is compared with the non-DL method presented in [37]. The method in [37] is a deterministic algorithm for attitude and trajectory estimation. It works with sequences of sonar images and it is capable of estimating pixel displacements between two sonar images with a subpixel accuracy. The pixel displacements are used to estimate the attitude and trajectory of the imaging sonar sensor. In this method, the same three parameters of the sensor motion are estimated (Δ_x, Δ_y and Δ_θ), so it is possible to make a direct comparison with the DL approach.

TABLE 7: Validation errors for the PoseNet-Normx10 when trained with images in polar and cartesian coordinates and PoseNet-Normx10 with a combination of polar and cartesian coordinates.

Approach	RMSE of motion estimation		
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(deg)*
	Images without noise		
Polar	2.84	1.00	0.052
Cartesian	3.36	1.08	0.051
Combined	2.60	1.07	0.047
	Images with noise		
Polar	6.54	3.39	0.170
Cartesian	9.75	3.33	0.111
Combined	4.99	2.27	0.086

For this comparison, only 100 out of the 8000 pairs of images from the validation data set were randomly chosen from the rocky field test set. The full test set is not used due to the high computational time required by the non-DL approach to compute the estimates. The comparison is performed using images with and without noise. PoseNet-Normx10 and its version trained with noisy images are used for the comparison. Both methods are implemented in Matlab and run on the same PC with Intel Core i5-6500 processor and 8GB of RAM.

The results are presented in Table 8. They show a better performance of the DL techniques in almost all the parameters, except in the forward/backward motion estimation using images with noise.

TABLE 8: Validation errors for the DL and non-DL techniques using 100 pairs of images.

Approach	RMSE of motion estimation		
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(deg)*
	Images without noise		
Non-DL	8.39	1.29	0.175
PoseNet-Normx10	3.33	1.08	0.054
PoseNet-Normx10wNoise	2.95	1.40	0.052
	Images with noise		
Non-DL	8.16	1.31	0.178
PoseNet-Normx10	7.03	3.25	0.165
PoseNet-Normx10wNoise	3.43	1.56	0.059

The non-DL approach presents a robust performance regardless the images are with or without noise. Overall, the best performance is achieved by the PoseNet-Normx10 trained with noisy images.

Furthermore, Table 9 shows the running time that each method requires to estimate from 100 pairs of images. It can

be seen that the running time is significantly higher for the non-DL technique compared to the DL network. The DL running time is 5.69s which can be split in two steps: the image concatenation (1.27s) and the estimation of the sonar motion (4.42s).

TABLE 9: Running times of the non-DL and DL methods using a data set of 100 pairs of images.

Method	Running time (s)
Non-DL	2568.74
DL	5.69

Therefore, the DL method can obtain estimates for each pair of images every 56.9ms. This suggests that this method could be implemented in real-time considering frame rates of an order of 20 fps (50 ms). However, a set of pretrained networks for different sonar parameters (such as the pitch angle and height) should be generated in advance.

I. DISCUSSION ABOUT MODIFICATIONS OF THE NETWORKS

In this section, we considered several approaches to training a network to estimate the motion of an underwater platform from acoustic images generated by a sonar. Five networks were implemented and validated, then the best networks were modified to improve their performance. The following points summarize the results obtained:

- The five networks were trained using quantized labels with images with and without noise. The best performance with noiseless images was provided by PoseNet and CNN-4PoseNet, whilst for images with noise, the best performance was provided by SfMNet.
- CNN-4PoseNet takes between 2 to 3 times longer to train, compared to the PoseNet and SfMNet.
- The CNN-1b and CNN-4b show a poorer performance compared to the other networks.
- The analysis of the estimation errors shows a high correlation between the errors of the sideways motion and the rotation.
- The SfMNet, PoseNet and CNN-4PoseNet show better results when trained using normalization of the labels rather than quantization. The best normalization is found to be within the range [-10, 10] with respect to the maximum motion values.
- The normalization results in a high improvement in the PoseNet performance when trained with noisy images and a similar performance to the quantization approach for noiseless images. The SfMNet shows similar results when trained with quantization and normalization of labels. The CNN-4PoseNet performance is observed to be worse with the normalization compared with the quantization. Therefore the PoseNet with normalization (PoseNet-Normx10) and the CNN-4PoseNet with quantization (CNN-4PoseNet-Qua) are selected for further modifications.
- A low level noise added to the images used for training the PoseNet-Normx10 and CNN-4PoseNet-Qua slightly increases the estimation errors when tested with noiseless images for both the networks. However, it considerably reduces the estimation error provided by PoseNet whereas the CNN-4PoseNet shows a similar performance when trained with noiseless and noisy images. Therefore PoseNet-Normx10 is selected for further optimization.
- Training the PoseNet-Normx10 with 3 concatenated images rather than 2 images, shows only slight improvement in the performance.
- A small dataset of wider images (higher aperture of the sonar in the azimuth dimension) was generated and used for training the PoseNet-Normx10. It shows estimation errors similar to the case of the original size images for noiseless images but an increase of the errors for noisy images. This can be due to larger image areas affected by noise but not carrying objects useful for motion estimation.
- Concatenating the images one besides the other (horizontally) or one above the other (vertically) is irrelevant for the networks performance. Both approaches show similar results.
- The loss function of the PoseNet-Normx10 was modified to penalize the cases when a parameter estimate falls outside the range [-10, 10]. The results show a similar performance when testing with noiseless images. With noisy images, it can be seen a small improvement for sideways and forward motion estimation but worse estimates for the rotation.
- The PoseNet-Normx10 trained with images in cartesian coordinates shows a slight increase in the estimation errors compared to the network trained with images in the original polar coordinates.
- The network that combines the images in polar and cartesian coordinates results in a smaller estimation error compared to the cases of using the polar or cartesian coordinates only.
- The PoseNet-Normx10 and PoseNet-Normx10wNoise have been compared with a non-DL method. Both DL networks achieve better estimation performance than the non-DL method.
- The computing time of the motion estimation is significantly reduced (by hundreds of times) when using the DL estimator instead of the non-DL method. The computing time for the DL approach is 56.9 ms for each pair of images, suggesting that this technique can be used in real-time applications.

V. TRAJECTORY ESTIMATION

A. TRAJECTORY ESTIMATION USING SYNTHETIC DATA

The already trained PoseNet-Normx10 network is used for estimating the trajectory of a sonar platform. Using the sonar simulator, a new data set was generated for validating the trajectory estimator. The scenario shown in Fig. 7 was used and

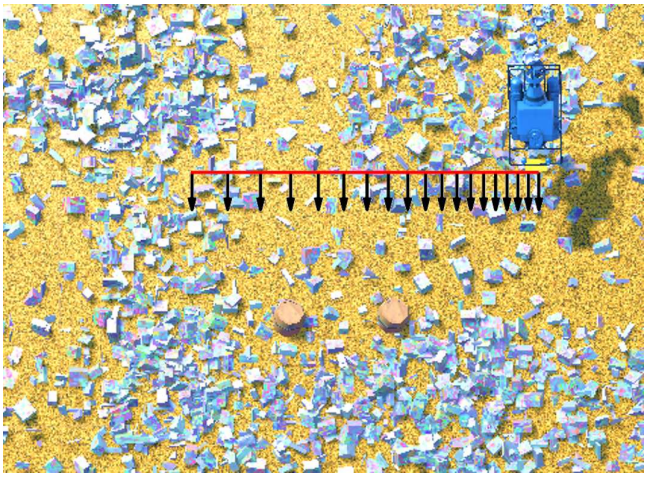


FIGURE 7: Scenario for testing and generating images for mosaics. The red line represents the trajectory of the platform. The black arrows show for every 50 images the sonar orientation. The sonar keeps looking forward when moving along the trajectory.

a trajectory with 904 noiseless images was created, keeping the same pitch angle and height from the seafloor. The motion of the sonar is in the sideway direction with a constant acceleration from 0.105 to 0.378 m/s along the trajectory. Every point of the trajectory is represented using global cartesian coordinates (x_i, y_i) and the platform orientation relative to the global scenario orientation (θ_i), which corresponds to 0° when facing in the y-axis direction; $i = 1, 2, 3, ..., M-1$, is the index for the image pairs and M is the number of images obtained on the trajectory. The first point of the trajectory is placed at the origin $(x_1 = 0, y_1 = 0, \theta_1 = 0)$.

The coordinates are assumed to be on a plane parallel to the seabed. The trajectory points are calculated as follows [59]:

$$\theta_{i+1} = \theta_i + \Delta_{\theta_i},$$
$$x_{i+1} = x_i + \Delta_{y_i}\sin(\theta_i + \Delta_{\theta_i}) + \Delta_{x_i}\cos(\theta_i + \Delta_{\theta_i}),$$
$$y_{i+1} = y_i + \Delta_{y_i}\cos(\theta_i + \Delta_{\theta_i}) - \Delta_{x_i}\sin(\theta_i + \Delta_{\theta_i}),$$

where $\Delta_{x_i}, \Delta_{y_i}$ and Δ_{θ_i} are the x and y-translation and z-rotation estimates for each pair of images, respectively.

An example mosaic built from the images using the motion estimates can be seen in Fig. 8c. The images are mapped to the global coordinates according to the reconstructed trajectory. To produce the mosaic, the pixel intensities at each point are interpolated from the image pixels. The intensity of a pixel in an image is averaged over images when they overlap. For clarity, only the pixels that correspond to 24 center beams of each image are used for interpolation, except for the first and last images. In Fig. 8b, it can be seen that the shape of the trajectory and orientation are similar to the ground truth provided by the simulator. However, the estimates tend to be smaller than the ground truth.

Fig. 9a shows estimated and ground truth trajectories from a data set of 904 images where the motion is performed in the forward direction with a constant acceleration corresponding to the speed from 0.105 to 0.378 m/s. From the shape of the estimated trajectory, it can be seen that in the forward direction the estimates are more accurate. The size of the estimated trajectory in the forward direction is almost the same as the ground truth.

In Fig. 9b, a forward moving trajectory is displayed for the case of acceleration and deceleration. During the generation of this data set, the sonar platform moves always forward and accelerates and decelerates with a minimum speed of 0.063 m/s and a maximum speed of 0.357 m/s. The recovered trajectory is close to the ground truth.

B. TRAJECTORY ESTIMATION USING THE SHIP'S HULL REAL DATA

The data set acquired by a real sonar (see subsection II-C) is used to validate the performance of the DL approach for attitude and trajectory estimation. The PoseNet-Normx10 network is trained using the simulated ship's hull described in subsection II-B. The network is trained three times applying different levels of noise on the images: noiseless, low level noise, which is the one used in subsection IV-C and high level noise, which is the same as that used for validation of the networks, since this is the level measured in the same real data set in [58]. A comparison of training with the three different levels of noise is presented in Table 10. The network trained with the high level of noise is selected for the trajectory estimation since it shows a better RMSE in the validation with noise, which is assumed to be similar to the real data set.

TABLE 10: Performance of the PoseNet-Normx10 network for the simulated ship's hull data set with different levels of noise

Approach	RMSE of motion estimation		
	Δ_x *(mm)*	Δ_y *(mm)*	Δ_θ *(deg)*
	Images without noise		
Noiseless images	2.99	1.63	0.045
Low level noise images	2.93	1.83	0.045
High level noise images	3.20	1.75	0.052
	Images with noise		
Noiseless images	4.41	2.72	0.067
Low level noise images	3.98	2.66	0.059
High level noise images	3.22	1.77	0.052

The already trained network is fed with the real data set to estimate the displacements between each pair of consecutive images. From the estimates the sonar trajectory is recovered as described in subsection V-A. Based on the shape and length of the obtained trajectory, the estimated trajectory seems to be shorter than expected in the sonar sideway direction. This assumption is based in the works [17], [37], which use the same real data set to estimate the trajectory of the sonar. The length of the ship's hull is estimated to be approximately 10 m. Therefore, all the estimates in the sideway direction are multiplied by a constant coefficient

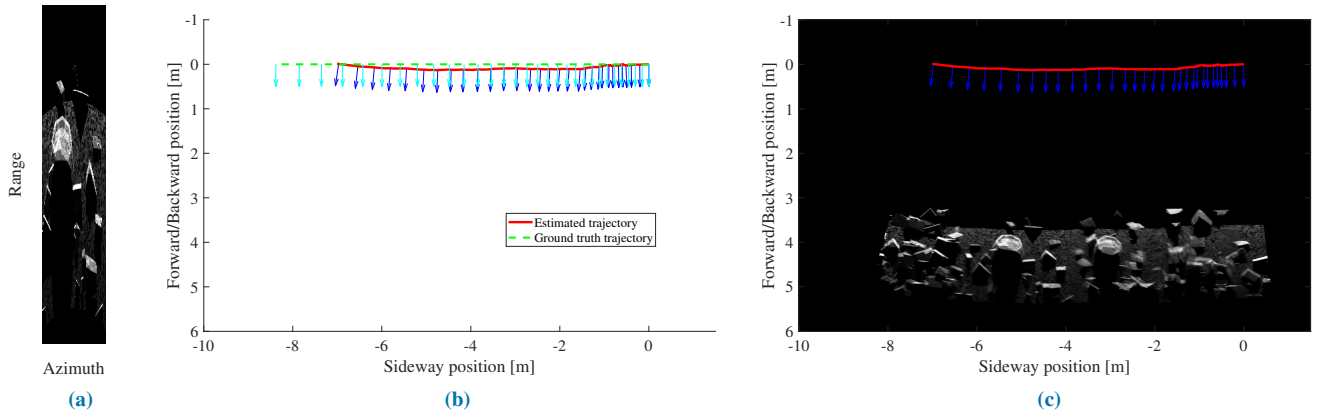


FIGURE 8: (a) Single sonar image in polar coordinates used for constructing the mosaic. (b) Mosaic of a data set of 904 simulated images. (c) Green lines with cyan arrows represent ground truth trajectory and pose of the sonar, respectively, whilst red line and dark blue arrows represent estimated trajectory and pose, respectively.

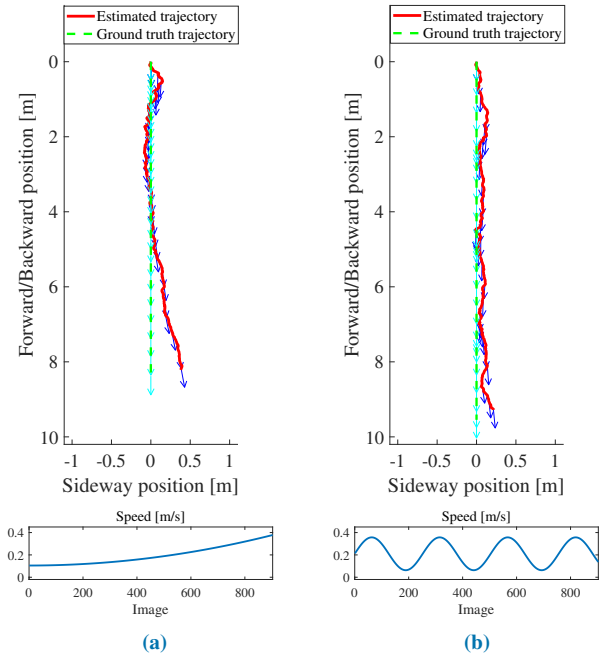


FIGURE 9: (a) Trajectory of a sonar that moves in forward direction with constant acceleration. (b) Trajectory of a sonar that moves in forward direction with acceleration and deceleration.

2.3 to provide a better match to the hull size. The need of such coefficient could be related to inaccuracies of setting the sonar parameters that are used by the simulator to create the training data set, like the height and sonar pitch angle.

By using the compensated trajectory, the images are merged into the mosaic shown in Fig. 10b. One single sonar image that is used to build the mosaic is presented in polar coordinates in Fig. 10a, corresponding to a fragment of the ship's keel, at the center-right of the mosaic. From the mosaic, it can be seen that motion estimates are reasonably accurate, with a few distortions in the image. Some sacrificial anodes and the keel are clearly recognizable.

C. TRAJECTORY ESTIMATION USING THE DAM INSPECTION REAL DATA

The data set described in subsection II-D was acquired by a real sonar during a dam inspection. The already trained PoseNet-Normx10wNoise is fed with the images from the data set to estimate the trajectory followed by the sonar. This network is selected due to similarity of objects in the images of the real data set with the rocky fields training set. However, the real data set presents significant differences that affect the estimate, such as the pitch angle of the sonar and its distance from the seabed. Therefore, the estimates obtained by the network are scale compensated to generate a more accurate trajectory. The compensation multiplies the estimates in the forward/backward motion and the sideways motion by a constant coefficient (4.0 and 9.0, respectively). The size of the sonar images is 1344×128 pixels and the FoV of the sonar is 30° along 128 beams in the azimuth dimension. To match the input size of the network, only pixels in a window of the size 512×96 in the center of each image are used for estimation. The estimates are used to generate the full sonar trajectory. Then the sonar images are merged into a mosaic (Fig. 11) by following the estimated trajectory. The dam wall can be seen as a continuous white line at the bottom of the mosaic. The data set contains sharp discontinuities, also the presence of fish in some images can cause low quality of the estimates and affect the full trajectory estimation. However, it can be seen that using a trained network, it is possible to produce a decent mosaic even if the sonar features of the training set are different. The mosaic is highly similar to the mosaic built in [60], which uses the deterministic method for the motion estimation followed by regularized P-splines for smoothing the trajectory.

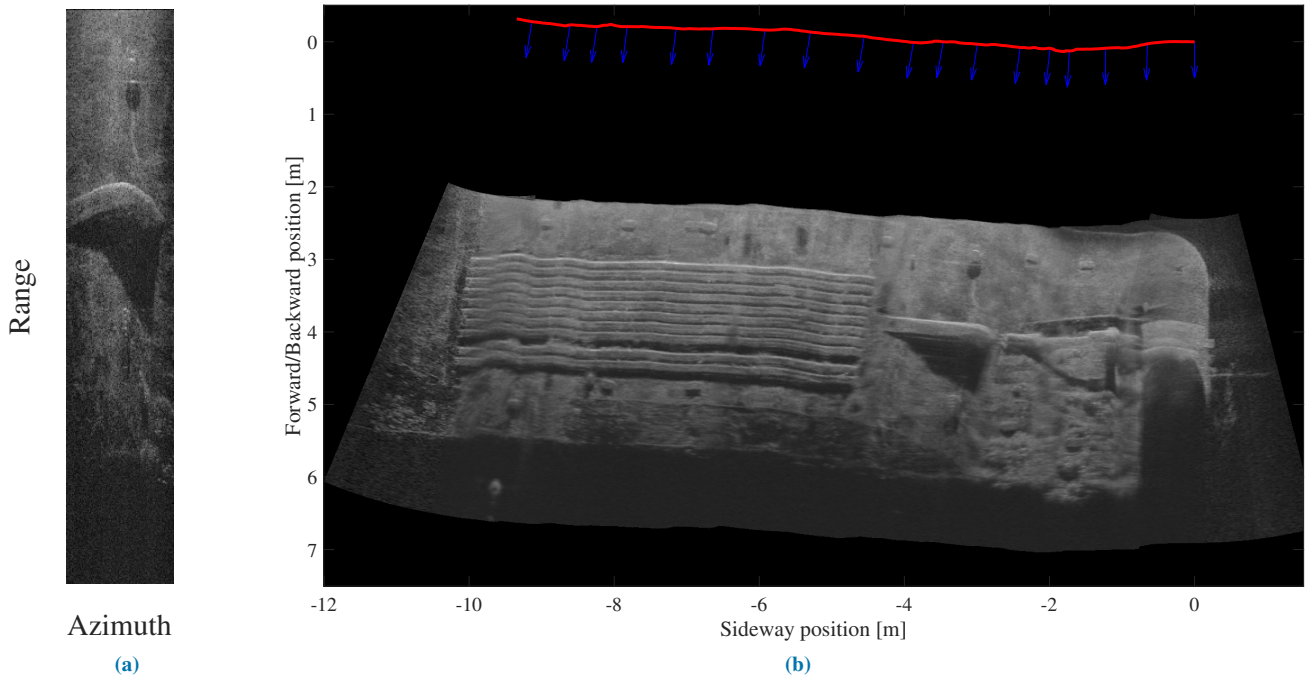


FIGURE 10: (a) Single real sonar image in polar coordinates used for constructing the mosaic. (b) Trajectory and mosaic obtained using 520 images of the ship's hull real data. The network used for estimation is the PoseNet-Normx10 trained with the simulated ship's hull data set with noise.

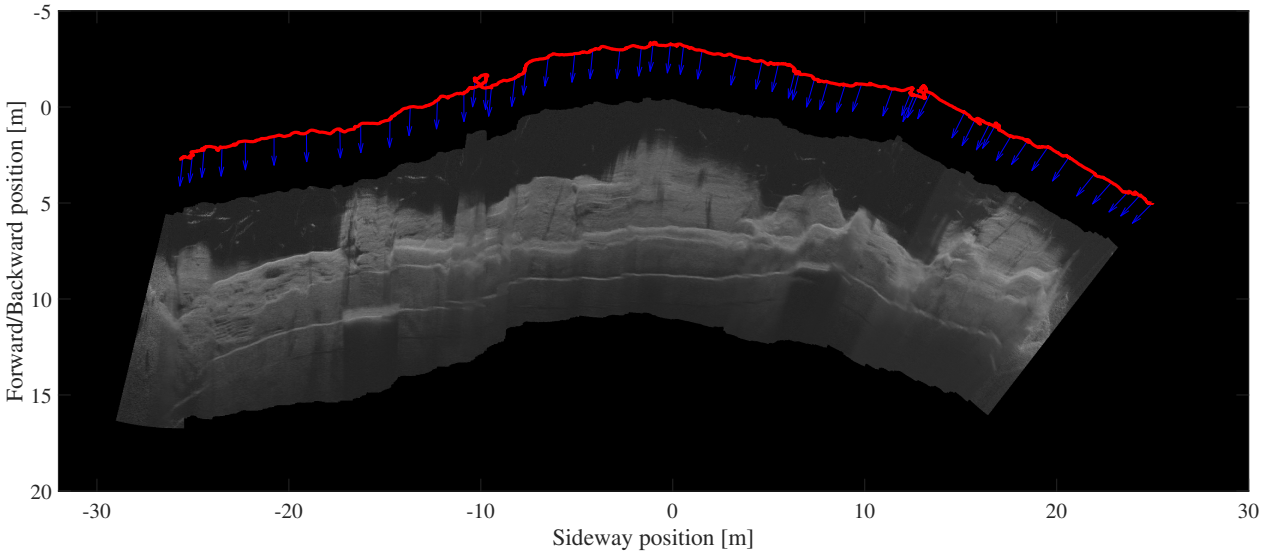


FIGURE 11: Trajectory and mosaic obtained using 1596 images of the dam inspection data set [55], [56]. The network used for estimation is PoseNet-Normx10wNoise trained with the simulated rocky field data set. The sonar sensor trajectory is shown in red and the attitude as a blue arrow every 30 images.

VI. CONCLUSIONS

The work presented in this paper is an attempt to use DL approaches for trajectory estimation using sonar images. The basis of this work is to use large volumes of synthetic images generated by a sonar simulator to train the DL networks, and then apply the already trained network to real data. The use of synthetic data provides the ground truth data needed to perform supervised learning and quantitatively validate the motion estimates. Several DL architectures and their modified versions are implemented, compared and presented

in this paper. The results obtained using different network architectures show that a millimeter accuracy for translation motion and below 0.1° for rotation motion can be achieved. Networks trained with simulated data and then used for estimation with real data sets can obtain reasonably good estimates even when the sonar features between the training set and the real data sets are different, by taking into account scale compensation parameters. The PoseNet architecture and its variations present the best results compared to the other networks. The PoseNet with normalized labels is applied to real sonar data sets, obtaining a good trajectory estimation that is used to construct mosaics by merging the images. The quality and realism of the simulated images used for training are important in the motion estimation applied to real data. For example, the use of the appropriate level of noise in the synthetic training images improved the accuracy of the trajectory estimation. The DL network obtained more accurate motion estimates using significantly lower computation time compared to the deterministic algorithm. However, the DL network requires retraining with a new data set if the sonar parameters change. One solution for this can be the use of multiple pretrained networks whose training data set is adjusted to different sonar parameters.

ACKNOWLEDGMENT

The authors would like to thank Prof. Y. Petillot, School of EPS, Heriot-Watt University and Dr. N. Hurtós for providing the ship's hull data set; Dr. Luis A. Conti, University of São Paulo and Acquest Subaquatic Geology and Geophysics for providing the dam wall inspection data.

REFERENCES

[1] H. Yu, W. Lu, D. Liu, Y. Han, and Q. Wu, "Speeding up gaussian belief space planning for underwater robots through a covariance upper bound," IEEE Access, vol. 7, pp. 121 961–121 974, 2019.

[2] H. Huang, J. Tang, B. Zhang, J. Chen, J. Zhang, and X. Song, "A novel nonlinear algorithm for non-gaussian noises and measurement information loss in underwater navigation," IEEE Access, vol. 8, pp. 118 472–118 484, 2020.

[3] J. González-García, A. Gómez-Espinosa, E. Cuan-Urquizo, L. G. García-Valdovinos, T. Salgado-Jiménez, and J. A. E. Cabello, "Autonomous underwater vehicles: Localization, navigation, and communication for collaborative missions," Applied Sciences, vol. 10, no. 4, p. 1256, 2020.

[4] Ø. Sture, P. Norgren, and M. Ludvigsen, "Trajectory planning for navigation aiding of autonomous underwater vehicles," IEEE Access, vol. 8, pp. 116 586–116 604, 2020.

[5] J. Zhang, Y. Han, C. Zheng, and D. Sun, "Underwater target localization using long baseline positioning system," Applied Acoustics, vol. 111, pp. 129–134, 2016.

[6] A. Mallios, P. Ridao, D. Ribas, M. Carreras, and R. Camilli, "Toward autonomous exploration in confined underwater environments," Journal of Field Robotics, vol. 33, no. 7, pp. 994–1012, 2016.

[7] D. Pick, E. Wolbrecht, M. Anderson, D. Edwards, and J. Canning, "Uncertainty analysis of ultra-short-and long-baseline localization systems for autonomous underwater vehicles," in OCEANS 2018 MTS/IEEE Charleston, 2018, pp. 1–6.

[8] W. Gao, Y. Zhang, and J. Wang, "Research on initial alignment and self-calibration of rotary strapdown inertial navigation systems," Sensors, vol. 15, no. 2, pp. 3154–3171, 2015.

[9] Q. Wu, K. Li, and J. Liu, "The asynchronous gimbal-rotation-based calibration method for lever-arm errors of two rotational inertial navigation systems," IEEE Access, vol. 7, pp. 4653–4663, 2018.

[10] F. Sun, W. Xu, and J. Li, "Enhancement of the aided inertial navigation system for an auv via micronavigation," in MTS/IEEE OCEANS 2010, Seattle.

[11] S. Caporale and Y. Petillot, "A new framework for synthetic aperture sonar micronavigation," arXiv preprint arXiv:1707.08488, 2017.

[12] S. Wirth, P. L. N. Carrasco, and G. O. Codina, "Visual odometry for autonomous underwater vehicles," in MTS/IEEE OCEANS 2013, Bergen, 2013, pp. 1–6.

[13] S. S. da Costa Botelho, P. Drews, G. L. Oliveira, and M. da Silva Figueiredo, "Visual odometry and mapping for underwater autonomous vehicles," in 6th IEEE Latin American Robotics Symposium (LARS), 2009, 2009, pp. 1–6.

[14] A. Kim and R. M. Eustice, "Real-time visual SLAM for autonomous underwater hull inspection using visual saliency," IEEE Transactions on Robotics, vol. 29, no. 3, pp. 719–733, 2013.

[15] M. Meireles, R. Lourenço, A. Dias, J. M. Almeida, H. Silva, and A. Martins, "Real time visual SLAM for underwater robotic inspection," in MTS/IEEE OCEANS 2014, St. John's, Newfoundland, 2014, pp. 1–5.

[16] R. Garcia and N. Gracias, "Detection of interest points in turbid underwater images," in IEEE OCEANS 2011, Santander, Spain, 2011, pp. 1–9.

[17] N. Hurtós, D. Ribas, X. Cufí, Y. Petillot, and J. Salvi, "Fourier-based registration for robust forward-looking sonar mosaicing in low-visibility underwater environments," Journal of Field Robotics, vol. 32, no. 1, pp. 123–151, 2015.

[18] F. Ferreira, D. Machado, G. Ferri, S. Dugelay, and J. Potter, "Underwater optical and acoustic imaging: A time for fusion? a brief overview of the state-of-the-art," in OCEANS 2016 MTS/IEEE Monterey, 2016, pp. 1–6.

[19] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, pp. 85–117, 2015.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, p. 436, 2015.

[21] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, "Exploring representation learning with CNNs for frame-to-frame ego-motion estimation," IEEE Robotics and Automation Letters, vol. 1, no. 1, pp. 18–25, 2015.

[22] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "SfM-Net: Learning of structure and motion from video," arXiv preprint arXiv:1704.07804, 2017, accessed: 2020-09-14. [Online]. Available: https://arxiv.org/pdf/1704.07804.pdf

[23] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1851–1858.

[24] Z. Yin and J. Shi, "GeoNet: Unsupervised learning of dense depth, optical flow and camera pose," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1983–1992.

[25] B. Teixeira, H. Silva, A. Matos, and E. Silva, "Deep learning for underwater visual odometry estimation," IEEE Access, vol. 8, pp. 44 687–44 701, 2020.

[26] H. Li and Y. Fan, "Non-rigid image registration using self-supervised fully convolutional networks without training data," in 15th IEEE International Symposium on Biomedical Imaging (ISBI 2018), 2018, pp. 1075–1078.

[27] B. D. de Vos, F. F. Berendsen, M. A. Viergever, M. Staring, and I. Išgum, "End-to-end unsupervised deformable image registration with a convolutional neural network," in Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support. Springer, 2017, pp. 204–212.

[28] D. Quan, S. Wang, M. Ning, T. Xiong, and L. Jiao, "Using deep neural networks for synthetic aperture radar image registration," in IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2016, pp. 2799–2802.

[29] S. Wang, D. Quan, X. Liang, M. Ning, Y. Guo, and L. Jiao, "A deep learning framework for remote sensing image registration," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 145, pp. 148–164, 2018.

[30] H. M. Keshk and X.-C. Yin, "Change detection in SAR images based on deep learning," International Journal of Aeronautical and Space Sciences, pp. 1–11, 2019.

[31] D. P. Williams and S. Dugelay, "Multi-view SAS image classification using deep learning," in MTS/IEEE OCEANS, Monterey, 2016, pp. 1–9.

[32] J. Ding, B. Chen, H. Liu, and M. Huang, "Convolutional neural network with data augmentation for SAR target recognition," IEEE Geoscience and Remote Sensing Letters, vol. 13, no. 3, pp. 364–368, 2016.

[33] P. Zhu, J. Isaacs, B. Fu, and S. Ferrari, "Deep learning feature extraction for target recognition and classification in underwater sonar images," in 56th IEEE Annual Conference on Decision and Control (CDC), 2017, pp. 2724–2731.

[34] S. Lee, "Deep learning of submerged body images from 2D sonar sensor based on convolutional neural network," in Proceedings of the IEEE Underwater Technology (UT) Conference, Busan, South Korea, 2017, pp. 1–3.
[35] X. Wang, J. Jiao, J. Yin, W. Zhao, X. Han, and B. Sun, "Underwater sonar image classification using adaptive weights convolutional neural network," Applied Acoustics, vol. 146, pp. 145–154, 2019.
[36] P. O. C. de Souza Ribeiro, M. M. dos Santos, P. L. J. Drews, and S. S. da Costa Botelho, "Forward looking sonar scene matching using deep learning," in 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017, pp. 574–579.
[37] B. T. Henson and Y. V. Zakharov, "Attitude-trajectory estimation for forward-looking multibeam sonar based on acoustic image registration," IEEE Journal of Oceanic Engineering, vol. 44, no. 3, pp. 753–766, 2018.
[38] T. A. Le, A. G. Baydin, R. Zinkov, and F. Wood, "Using synthetic data to train neural networks is model-based reasoning," in IEEE International Joint Conference on Neural Networks (IJCNN), 2017, pp. 3514–3521.
[39] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2758–2766.
[40] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4040–4048.
[41] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 969–977.
[42] A. Rozantsev, V. Lepetit, and P. Fua, "On rendering synthetic images for training an object detector," Computer Vision and Image Understanding, vol. 137, pp. 24–37, 2015.
[43] L. Taylor and G. Nitschke, "Improving deep learning using generic data augmentation," arXiv preprint arXiv:1708.06020, 2017, accessed: 2020-09-14. [Online]. Available: https://arxiv.org/pdf/1708.06020.pdf
[44] E. Richardson, M. Sela, and R. Kimmel, "3d face reconstruction by learning from synthetic data," in 2016 Fourth International Conference on 3D Vision (3DV). IEEE, 2016, pp. 460–469.
[45] A. Kortylewski, A. Schneider, T. Gerig, B. Egger, A. Morel-Forster, and T. Vetter, "Training deep face recognition systems with synthetic data," arXiv preprint arXiv:1802.05891, 2018, accessed: 2020-03-05. [Online]. Available: https://arxiv.org/pdf/1802.05891.pdf
[46] X. Zhang, Y. Fu, A. Zang, L. Sigal, and G. Agam, "Learning classifiers from synthetic data using a multichannel autoencoder," arXiv preprint arXiv:1503.03163, 2015, accessed: 2020-09-14. [Online]. Available: https://arxiv.org/pdf/1503.03163.pdf
[47] G. Georgakis, A. Mousavian, A. C. Berg, and J. Kosecka, "Synthesizing training data for object detection in indoor scenes," arXiv preprint arXiv:1702.07836, 2017, accessed: 2020-09-14. [Online]. Available: https://arxiv.org/pdf/1702.07836.pdf
[48] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2315–2324.
[49] F. Zhan, H. Zhu, and S. Lu, "Scene text synthesis for efficient and effective deep network training," arXiv preprint arXiv:1901.09193, 2019, accessed: 2020-09-14. [Online]. Available: https://arxiv.org/pdf/1901.09193.pdf
[50] J. E. Almanza-Medina, B. T. Henson, and Y. V. Zakharov, "Imaging sonar simulator for assessment of image registration techniques," in MTS/IEEE OCEANS 2019, Seattle, 2019.
[51] DIDSON 300 Standard Version Specifications, Sound Metrics Corp.
[52] K. Perlin, "Improving noise," in ACM Transactions on Graphics (TOG), vol. 21, no. 3. ACM, 2002, pp. 681–682.
[53] Hovering Autonomous Underwater Vehicle Specifications, Bluefin Robotics Corporation, 2015, accessed: 2020-09-14. [Online]. Available: https://gdmissionsystems.com/-/media/General-Dynamics/Maritime-and-Strategic-Systems/Bluefin/PDF/Bluefin-HAUV-Datasheet.ashx.
[54] J. Vaganay, M. Elkins, S. Willcox, F. Hover, R. Damus, S. Desset, J. Morash, and V. Polidoro, "Ship hull inspection by hull-relative navigation and control," in Proceedings of OCEANS 2005 MTS/IEEE, 2005, pp. 761–766.
[55] L. Conti, M. Rodriques, and B. Hanot, "Hydroelectric power plant inspections," Hydro International Magazine, vol. 20, no. 4, pp. 16–19, 2016.
[56] "Acquest websiste," https://www.acquest.com.br, accessed: 2021-01-28.
[57] ARIS Explorer 3000 Standard Version Specifications, Sound Metrics Corp., accessed: 2021-01-28. [Online]. Available: http://www.soundmetrics.com/Products/ARIS-Sonars/ARIS-Explorer-3000/016621-A-ARIS-Explorer-3000-Specifications.
[58] B. T. Henson, "Image registration for sonar applications," Ph.D. dissertation, University of York, 2017, accessed: 2020-09-14. [Online]. Available: http://etheses.whiterose.ac.uk/19536/1/thesis.pdf
[59] S. Marschner and P. Shirley, Fundamentals of Computer Graphics. CRC Press, 2015.
[60] B. Henson and Y. Zakharov, "Estimating attitude and trajectory of forward looking imaging sonar using inter-frame registration," in International Conference and Exhibition on Underwater Acoustics (UACE), Skiathos, Greece, 2017.

JOSÉ E. ALMANZA-MEDINA (M'19) received his M.Sc degree in electronics and telecommunications in 2015 from the Ensenada Center for Scientific Research and Higher Education (CICESE). From 2009 to 2013, he worked in Softtek IT company, Mexico, as a software engineer and project manager. He is currently working toward a Ph.D degree in electronic engineering in the Communication Research Group, Department of Electronic Engineering at the University of York, UK. His research interests include signal and image processing and underwater acoustics.

BENJAMIN HENSON (M'17) received his M.Eng. degree in electronic engineering in 2001 from University of York, U.K.. From 2002 to 2008, he worked as an engineer for Snell & Wilcox Ltd. designing broadcast equipment. From 2008 to 2009, he worked for SRD Ltd. on imaging sonar designs. He received an M.Sc degree in Natural Computation in 2011 from University of York, U.K.. Then, from 2011 to 2013, he worked on laser measurement equipment for Renishaw plc. He received his PhD. degree in electronic engineering in 2018 from the Communication Technologies Research Group, University of York U.K., where he is currently working as a Research Associate in the Department of Electronic Engineering. His interests include signal and image processing, acoustics.

YURIY ZAKHAROV (M'01 - SM'08) received the M.Sc. and Ph.D. degrees in electrical engineering from the Power Engineering Institute, Moscow, Russia, in 1977 and 1983, respectively. From 1977 to 1983, he was with the Special Design Agency in the Moscow Power Engineering Institute. From 1983 to 1999, he was with the N. N. Andreev Acoustics Institute, Moscow. From 1994 to 1999, he was with Nortel as a DSP Group Leader. Since 1999, he has been with the Communications Research Group, University of York, UK, where he is currently a Reader in the Department of Electronic Engineering. His research interests include signal processing, communications, and underwater acoustics.